AVTCamera
ActiveX Control

User Manual

Allied Vision Technologies GmbH
Taschenweg 2a
D-07646 Stadtroda - Germany

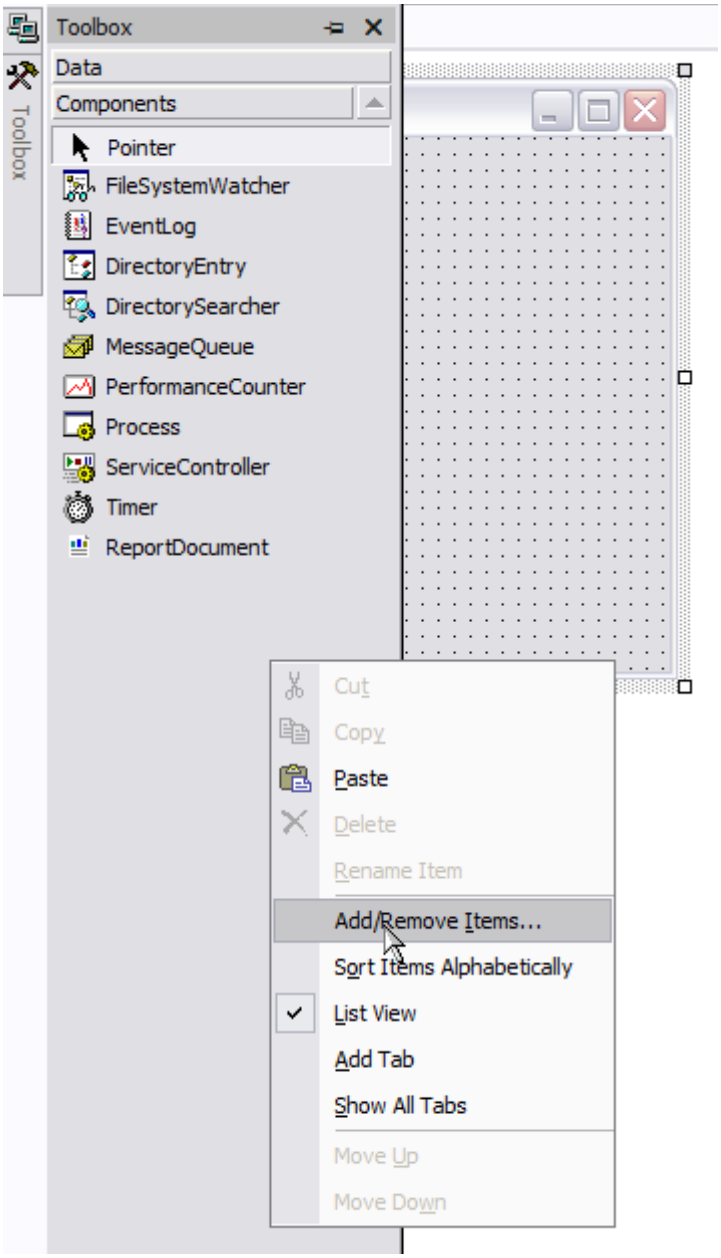
INTRODUCTION **3**
INSTALLATION **3**
FUNCTIONS AND PROPERTIES **5**
EVENTS **9**
SAMPLES **10**
INITIALISING 10
SAVE A IMAGE 10
DIRECT BUFFER ACCESS 11
SAMPLE PROJECTS 12

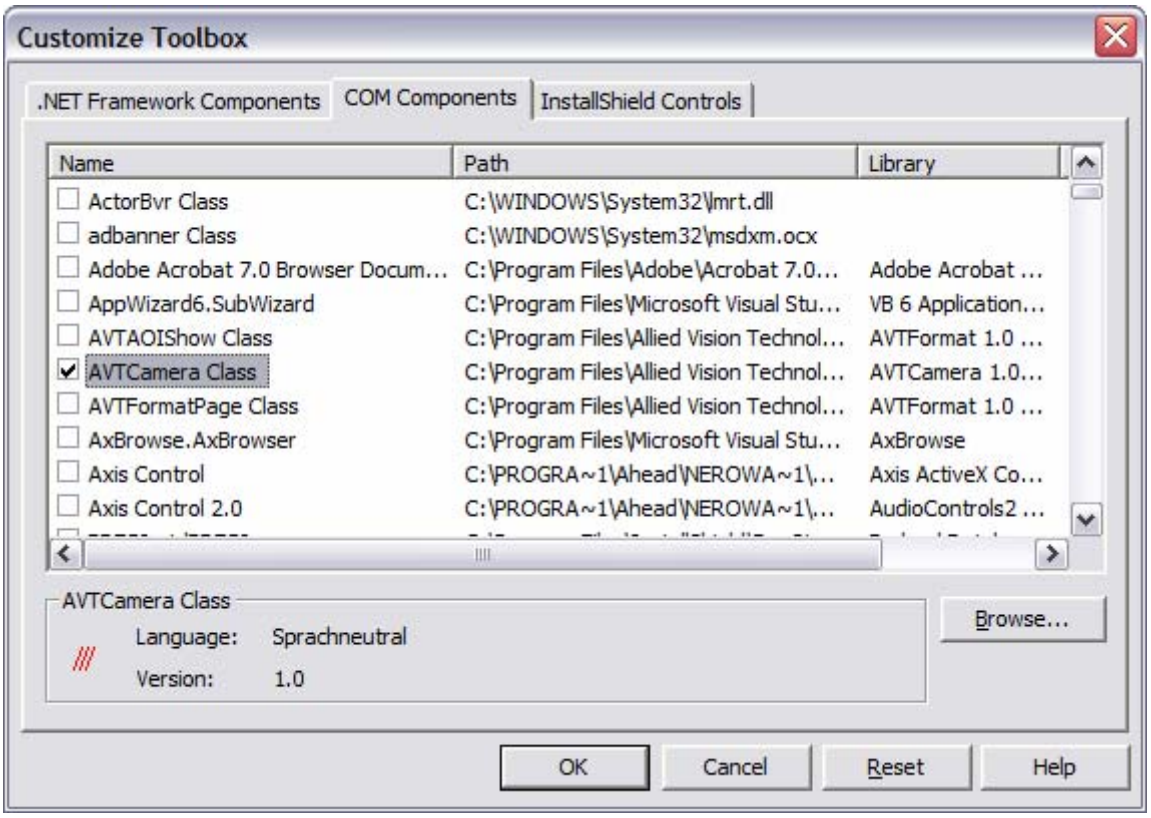
Introduction

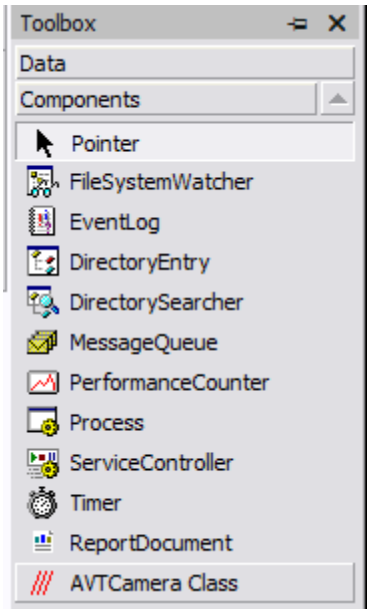
Based on the fundamentals of working with ActiveX Controls this manual only explain special functions, properties and events of the control.

Installation

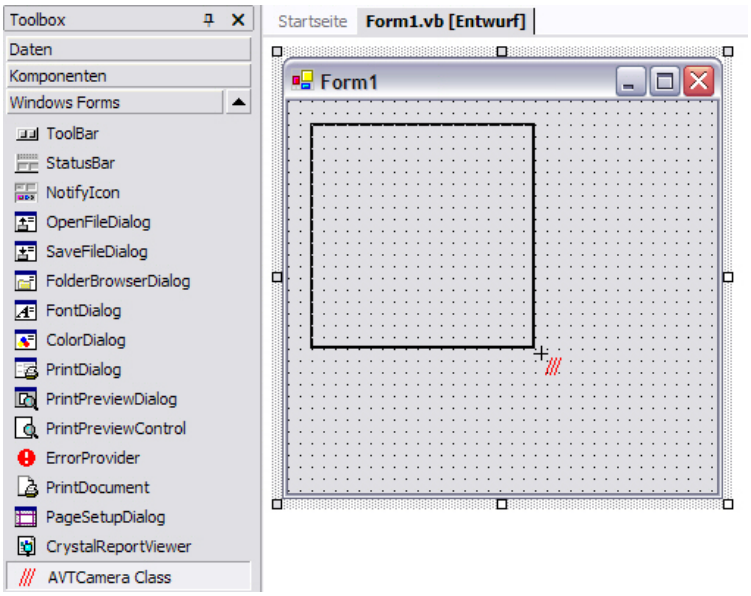
The ActiveX control is installed together with the package and registered on the computer. For inserting the control into a project add it first in the IDE toolbox.







Now it can use via Drag and Drop like all other controls.



Functions and Properties

C++: `HRESULT InitCameraList(void);`

VB.NET: `InitCameraList()`

Initialize the control and build a list of all available cameras.

C++: `HRESULT CameraCount([out, retval] LONG* pVal);`

VB.NET: `ReadOnly CameraCount As Integer`

Property; ReadOnly; counts all cameras in the list

C++:
```
HRESULT GetCameraName([in] LONG index,
[out,retval] BSTR* name);
```

VB.NET: `GetCameraName(index as Integer) as String`

Gets the name of the camera with the given index

C++: `HRESULT SelectCamera([in] LONG index);`

VB.NET: `SelectCamera(index as Integer)`

Selects a camera for further use with the control

C++: `HRESULT Start(void);`

VB.NET: `Start()`

Start image capture

C++: `HRESULT Stop(void);`

VB.NET: `Stop()`

Stop image capture

C++: `HRESULT BW([out, retval] VARIANT_BOOL* pVal);`

VB.NET: `ReadOnly BW as Boolean`

Property; ReadOnly; it indicates whether the stream is colored or b/w

C++: `HRESULT CameraProperties([in] LONG hWnd);`

VB.NET: `CameraProperties(hWnd as Integer)`

Show the DirectShow-Format-Properties-Dialog; the parameter can be a handle from a parent window

C++: `HRESULT FormatProperties([in] LONG hWnd);`

VB.NET: `FormatProperties(hWnd as Integer)`

Show the Camera-Properties-Dialog; the parameter can be a handle from a parent window

C++: `HRESULT FrameResize([out, retval] VARIANT_BOOL* pVal);`

VB.NET: `FrameResize as Boolean`

Property; Read/Write; control, if the image is scaled to the control size

C++: `HRESULT Width([out, retval] LONG* pVal);`

VB.NET: `ReadOnly Width as Integer`

Property; ReadOnly; width of the image in pixel

C++: `HRESULT Height([out, retval] LONG* pVal);`

VB.NET: `ReadOnly Height as Integer`

Property; ReadOnly; height of the image in pixel

C++: `HRESULT SavePicture(BSTR fileName);`

VB.NET: `SavePicture(fileName as String)`

Save the image with the given path name

C++:
```
HRESULT Format([in] LONG width, LONG height,
[in] VARIANT_BOOL color);
```

VB.NET:
```
Format(width as Integer, height as Integer,
color as Boolean)
```

Set the stream format. The values for with and height and colour coding must correspond with valid camera parameters. Call the function only, when image capturing is stopped.

C++: `HRESULT SetValue(ULONG address, ULONG value);`

VB.NET:
```
SetValue(address as System.UInt32,
value as System.UInt32)
```

Allow access to all camera registers.

C++:
```
HRESULT GetValue([in] ULONG address,
[out,retval] ULONG* value);
```

VB.NET: `GetValue(address as System.UInt32) as System.UInt32`

Allow access to all camera registers.

C++: `HRESULT GetColorCount([out,retval] LONG* colors);`

VB.NET: `GetColorCount() as Integer`

Count all available colour formats

C++:
```
HRESULT GetColor([in] LONG index,
[out,retval] BSTR* color);
```

VB.NET: `GetColor(index as Integer) as String`

Return a description of the colour format with the given index

C++: `HRESULT GetResolutionCount([in] LONG color, [out,retval] LONG* count);`

VB.NET: `GetResolutionCount(color as Integer) as Integer`

Return the count of resolutions to a given colour format

C++: `HRESULT GetResolution(LONG color, LONG index, [out,retval] BSTR* resolution);`

VB.NET: `GetResolution(color as Integer, index as Integer) as String`

Return a description of a resolution

C++: `HRESULT GetMaxFPS(LONG color, LONG index, [out,retval] DOUBLE* fps);`

VB.NET: `GetMaxFPS(color as Integer, index as Integer) as Double`

Return the maximal possible frame rate to a resolution and colour format

C++: `HRESULT SetFormatFPS(LONG color, LONG resolution, DOUBLE fps);`

VB.NET: `SetFormatFPS(color as Integer, resolution as Integer, fps as Double)`

Select a certain format. The possible index values can be determined with the before described functions. This function is possible only if the image capturing is stopped.

The following functions allow the direct access to the buffer with the image data.

C++: `HRESULT BufferAccess([out, retval] VARIANT_BOOL* pVal);`

VB.NET: `BufferAccess as Boolean`

Property;Read/Write; enable the direct buffer access; with enabled access the performance is a little lower

C++: `HRESULT GetBuffer([out,retval] BYTE** buffer);`

VB.NET: `GetBuffer() as System.IntPtr`

Get a pointer to the buffer with the image data. The control allocates new memory and copy the data. The calling application must free the memory!

C++: `HRESULT GetBufferSize([out,retval] LONG* size);`

VB.NET: `GetBufferSize() as Integer`

Get the size of the buffer in bytes.

C++: `HRESULT GetBitmapHeader([out,retval] BYTE** header);`

VB.NET: `GetBitmapHeader() as System.IntPtr`

Get a pointer to a buffer with a valid bitmap header. The control allocates new memory and copy the data. The calling application must free the memory!

C++: `HRESULT GetBitmapHeaderSize([out,retval] LONG* size);`

VB.NET: `GetBitmapHeaderSize() as Integer`

Get the size of the buffer for the bitmap header in bytes.

C++: `HRESULT GetBitmapFileHeader([out,retval] BYTE** buffer);`

VB.NET: `GetBitmapFileHeader() as System.IntPtr`

Get a pointer to a buffer with a valid bitmap file header. The control allocates new memory and copy the data. The calling application must free the memory!

C++: `HRESULT GetBitmapFileHeaderSize([out,retval] LONG* size);`

VB.NET: `GetBitmapFileHeaderSize() as Integer`

Get the size of the buffer for the bitmap header in bytes.

Events

`HRESULT SavePicture([in] BSTR file);`

Fired, when a file is saving

`HRESULT NextPicture([in] DOUBLE time);`

Fired, after a new image is captured

`HRESULT SaveOk(void);`

Fired, after a file is saved or the direct access to the image data is possible

Samples

All functions are described by the provided examples. Here some fundamental proceedings.

Initialising

```
Dim cameraCounter As Integer
Dim i As Integer
Dim name As String

AxAVTCamera1.InitCameraList()
cameraCounter = AxAVTCamera1.CameraCount

CameraList.Items.Clear()

For i = 0 To cameraCounter - 1 Step 1
      name = AxAVTCamera1.GetCameraName(i)
      CameraList.Items.Add(name)
Next

If CameraList.Items.Count > 0 Then
      CameraList.SelectedIndex = 0
End If
```

Save a Image

The picture stored with click on a Button. The control waits, until the camera supplies the next picture. Then the picture is stored and the appropriate Event is released.
The application evaluates the parameters of the Events and calls a registered application for the displaying of the picture.

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As ⮌
 System.EventArgs) Handles Button5.Click
        AxAVTCamera1.SavePicture("Test.bmp")
End Sub

Private Sub AxAVTCamera1_SavePictureEvent(ByVal sender As Object, ByVal e ⮌
As AxAVTCameraLib._IAVTCameraEvents_SavePictureEvent) Handles ⮌
AxAVTCamera1.SavePictureEvent
      Dim pInfo As New ProcessStartInfo
      pInfo.FileName = e.file
      pInfo.UseShellExecute = True
       Dim p As Process = Process.Start(pInfo)
End Sub
```

Direct buffer access

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As ⮌
System.EventArgs) Handles MyBase.Load
      …
      Me.AxAVTCamera1.SelectCamera(0)
      Me.AxAVTCamera1.BufferAccess = True
      …

Private Sub AxAVTCamera1_SaveOk(ByVal sender As Object, ByVal e As ⮌
System.EventArgs) Handles AxAVTCamera1.SaveOk
      ‘ if you try to access the buffer in this event, it's a bad idea
      ' because the next event can come quickly
      ' and the framework mix the parameters of the events
      ' so you get something strange with the Marshal interface...

      Console.WriteLine("SaveOk")
      Me.Button5.Enabled = True
End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As ⮌
System.EventArgs) Handles Button5.Click
      ' the pointer to the buffer for
      ' the picture data (BGR)
      ' the bitmap header
      ' the bitmap file header
      Dim picBuffer As System.IntPtr
      Dim headerBuffer As System.IntPtr
      Dim fileBuffer As System.IntPtr

      ' the size of buffers
      Dim picLen As Integer
      Dim headerLen As Integer
      Dim fileLen As Integer

      ' get the size
      picLen = Me.AxAVTCamera1.GetBufferSize()
      headerLen = Me.AxAVTCamera1.GetBitmapHeaderSize()
      fileLen = Me.AxAVTCamera1.GetBitmapFileHeaderSize()

      'reserve the own buffers
      Dim myPicBuffer(picLen) As Byte
      Dim myHeaderBuffer(headerLen) As Byte
      Dim myFileBuffer(fileLen) As Byte

      ' get the buffers
      picBuffer = Me.AxAVTCamera1.GetBuffer()
      headerBuffer = Me.AxAVTCamera1.GetBitmapHeader()
      fileBuffer = Me.AxAVTCamera1.GetBitmapFileHeader()

      ' is the pointer valid ?
      If picBuffer.ToInt32() <> 0 Then
             ' copy the data in a way, unmanaged code understand it
             ' I love the Marshal Class
             Marshal.Copy(picBuffer, myPicBuffer, 0, picLen)
             ' never forget to free the buffer
             Marshal.FreeCoTaskMem(picBuffer)
```

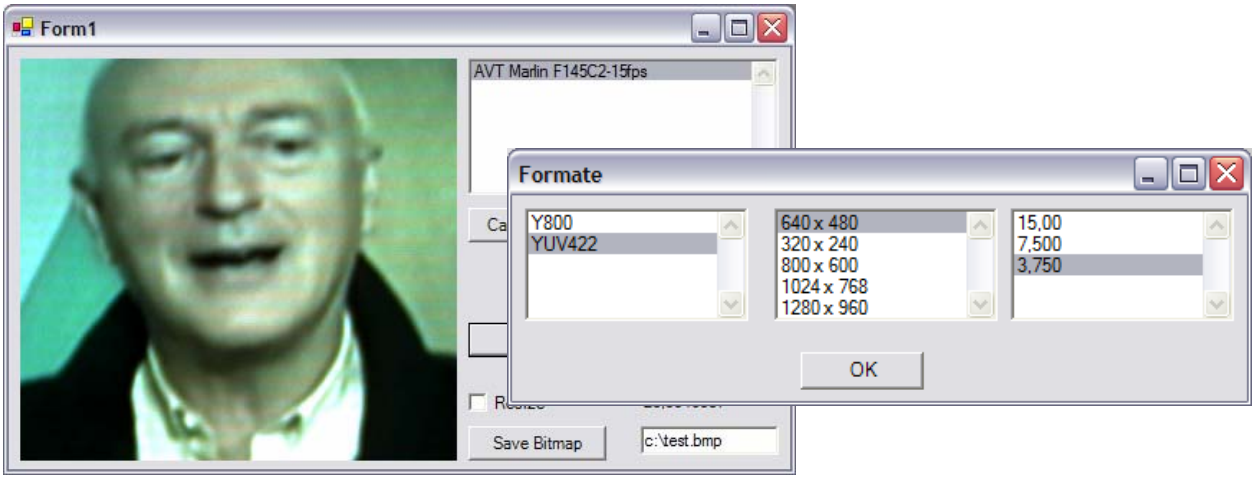
```
      End If

      ' same procedure as...
      If headerBuffer.ToInt32() <> 0 Then
            Marshal.Copy(headerBuffer, myHeaderBuffer, 0, headerLen)
            Marshal.FreeCoTaskMem(headerBuffer)
      End If

      ' ...
      If fileBuffer.ToInt32() <> 0 Then
            Marshal.Copy(fileBuffer, myFileBuffer, 0, fileLen)
            Marshal.FreeCoTaskMem(fileBuffer)
      End If

      ' open a file
      Dim fs As New FileStream("temp.bmp", FileMode.Create)
      Dim w As New BinaryWriter(fs)

      ' and write the buffer
      ' (if you forget the len parameter, VB add a 0), so...
      w.Write(myFileBuffer, 0, fileLen)
      w.Write(myHeaderBuffer, 0, headerLen)
      w.Write(myPicBuffer, 0, picLen)

      ' and close it
      w.Close()
      fs.Close()
End Sub
```

Sample projects

The following sample projects are installed by the package:

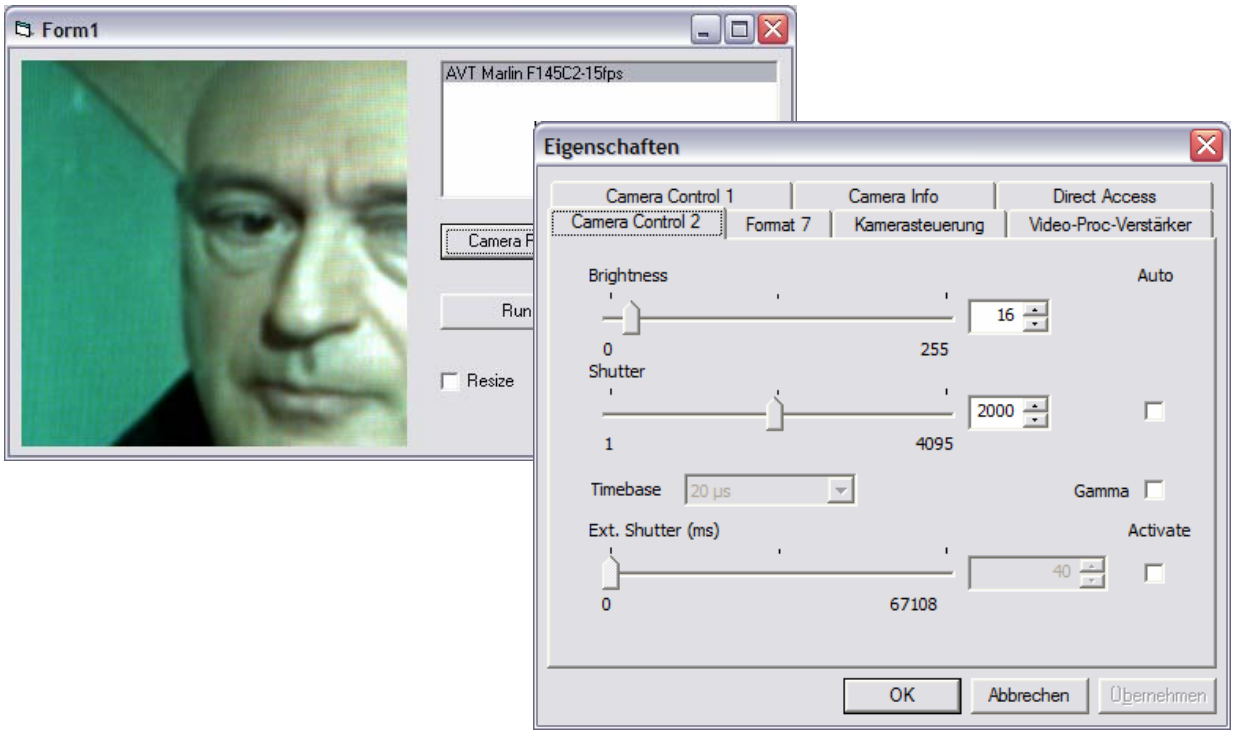
`C:\Program Files\Allied Vision Technologies\DirectFire Package\avtcamerademo`

A VB.NET Sample, which shows basic proceedings, like selection of a camera, show property pages, save a picture and build an own format dialog.



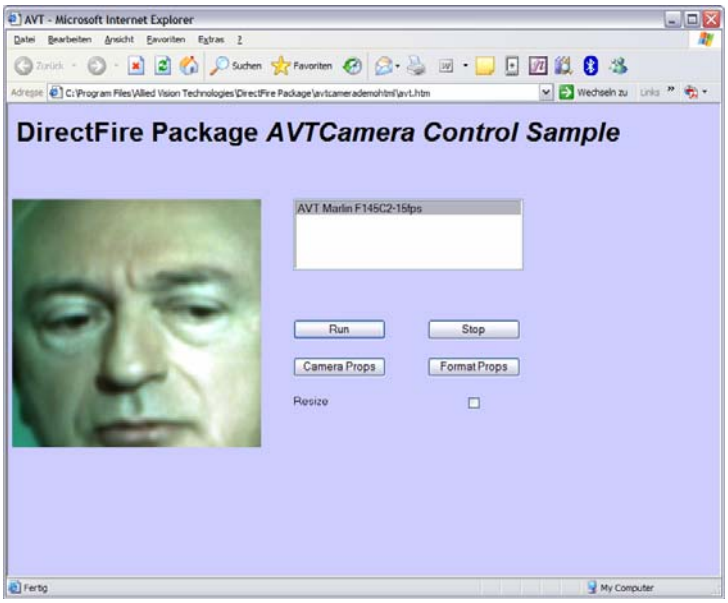
C:\Program Files\Allied Vision Technologies\DirectFire Package\avtcamerademo6

A VB6 Project that shows the same features.



C:\Program Files\Allied Vision Technologies\DirectFire Package\avtcamerademohtml

Show the usage of the control in a HTML and Scripting environment.



C:\Program Files\Allied Vision Technologies\DirectFire Package\pictureaccess

This VB.NET sample shows basic functions of direct buffer access.